# Projet QRCODE

## **Partie 2 :**

Dans cette partie je devais réaliser la partie masquage du QRCODE avec le meilleur choix de masque.

### **Génération des masks :**

Tout d’abord je me suis penché sur la réalisation des différents masques à l’aide de la formule donner par la norme ISO 18004/2015

Une image contenant texte

Description générée automatiquement

En vérifiant bien que le cahier des charges à bien été respecter sur la génération du mask. Pour rappel le masque ne doit pas être généré sur le finder pattern, timing pattern et Version Information

Une image contenant texte, mots croisés, carrelé

Description générée automatiquement  
Dans notre cas le cahier des charges est bien respecté.

aaaa

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | **void** **genere\_QRmask**(**unsigned** **char** qrmask[NB\_MODULE][NB\_MODULE],**int** no\_masque)  {  printf("**\n** donnee Utile : **\t**Numero du masque :%d **\t** Nombre de module : %d **\n**", no\_masque, NB\_MODULE);  //Afin d'avoir un aperçu sur la console des valeurs transmissent dans la boucle.  **int** i,j, finder =**9**;  //Envoie de donnée dans un tableaux, avec la formules  //des masks  **for**(i=**1**; i<NB\_MODULE; i++)  {  **for**(j=**1**; j<NB\_MODULE; j++)  {    **switch** (no\_masque)  {  **case** **0**: //Si le masque 0 est sélectionner  **if**(i%**2**==**0**)  qrmask[i][j]=NOIR;  **if**(i%**2**!=**0**)  qrmask[i][j]=BLANC;  **break**;  **case** **1**: //Si le masque 1 est sélectionner  **if**((i/**2**+j/**3**)%**2**==**0**)  qrmask[i][j]=NOIR;  **if**((i/**2**+j/**3**)%**2**==**0**)  qrmask[i][j]=BLANC;  **break**;  **case** **2**: //Si le masque 2 est sélectionner  **if**(((i\*j)%**2**+(i\*j)%**3**)%**2**==**0**)  qrmask[i][j]=NOIR;  **if**(((i\*j)%**2**+(i\*j)%**3**)%**2**!=**0**)  qrmask[i][j]=BLANC;  **break**;  **case** **3**: //Si le masque 3 est sélectionner  **if**(((i+j)%**2**+(i\*j)%**3**)%**2**==**0**)  qrmask[i][j]=NOIR;  **if**(((i+j)%**2**+(i\*j)%**3**)%**2**!=**0**)  qrmask[i][j]=BLANC;  **break**;  }  **if**(i< finder && j<finder ||j==**0** && i<NB\_MODULE ||i==**0** && j<NB\_MODULE)  qrmask[i][j]=BLANC;  }  }  } |

Voici le code nous permettant

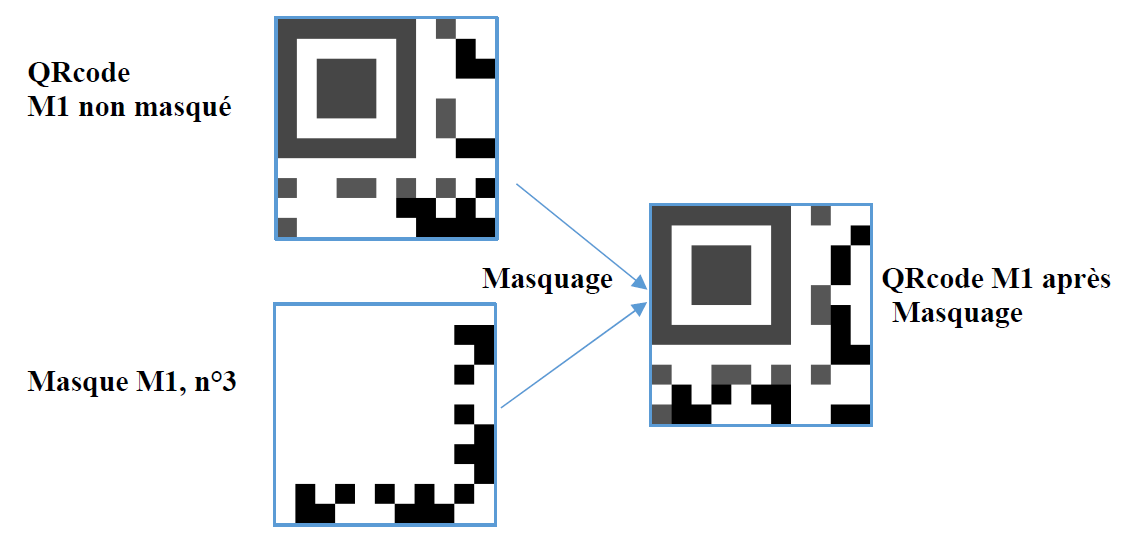
De générer les différents masques

### Réalisation du XOR

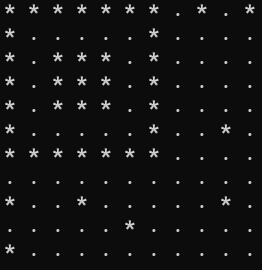
Maintenant nous devons réaliser un XOR afin de combiner le masque généré avec le qrcode.

On devait respecté comme contrainte que si le module du masque est noir et que celui du qrcode est blanc, alors le qrcode final prendra comme module la couleur noir. En Revanche si le masque et le qrcode sont tout deux de couleur noir, alors le qrcode final aura pour module la couleur blanc.

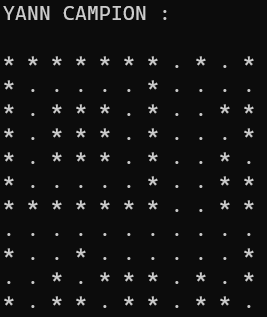
Une image contenant texte

Description générée automatiquement

Pattern généré depuis la partie 0.

Masque généré

Quelques valeurs de test afin de voir si le xor fonctionne bien correctement lors de son exécution

Une fois la fonction du XOR effectuer nous obtenons ainsi le résultat suivant :

Le Xor réalise donc bien le cahier des charges.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | **void** **xor\_QRcode\_QRmask**(**unsigned** **char** qrcode[NB\_MODULE][NB\_MODULE],**const** **unsigned** **char** qrmask[NB\_MODULE][NB\_MODULE])  {  **int** i,j;  //appel des deux tableaux  **for**(i=**0**; i<NB\_MODULE; i++)  {  **for**(j=**0**; j<NB\_MODULE; j++)  {  //mettre les cases noirs du masque sur les cases blanches du qrcode  **if**(qrmask[i][j]==NOIR && qrcode[i][j]==BLANC)  qrcode[i][j]=NOIR;  //sinon si le masque a les mêmes cases que le qrcode en noir mettre dans le  // qrcode final la case en blanc  **else** **if** (qrmask[i][j]==NOIR && qrcode[i][j]==NOIR)  qrcode[i][j]=BLANC;  }  }  } |

### Génération du score pour le masque

Je devais faire un score, permettant de noter le masque choisi après exécution du XOR. Pour cela il devait compter le nombre de case noir étant sur la dernière ligne, et la dernière colonne du qrcode, puis calculer le score du masque à l’aide de la formule de la norme ISO18004/2015 (p54)

Une image contenant texte

Description générée automatiquement

Notre score est donc correct avec le masque choisi.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | **int** **score\_masquage\_QRcode**(**const** **unsigned** **char** qrcode[NB\_MODULE][NB\_MODULE])  {  **int** som\_1=**0**, som\_2=**0**, i,j;  **int** score=**0**;  **for**(i=**0**; i<NB\_MODULE; i++)  {  **for**(j=**0**; j<NB\_MODULE; j++)  {  //j==10 Dernière colone du qrcode+Comptage case noir  //dans cette colone  **if**(j==**10** && qrcode[i][j]==NOIR )  som\_1=som\_1+**1**;  //i==10 Dernière ligne du QRcode+comptage case noir  **if**(i==**10** && qrcode[i][j]==NOIR)  som\_2=som\_2+**1**;  //Calcule du score final  **if**(som\_1<=som\_2)  score=som\_1\***16**+som\_2;  **if**(som\_1>som\_2)  score=som\_2\***16**+som\_1;  }  }  printf("score = %d",score);  //Renvoie dans le test unitaire 2 la valeur du score  **return** score;  } |

### Choix du meilleur masque pour une génération optimal du qrcode

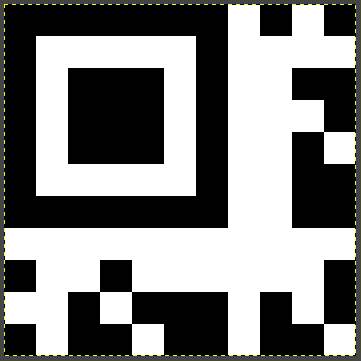
Une image contenant texte

Description générée automatiquementPour cela on devait utiliser le masque nous donnant le score le plus important. J’ai généré les 4 masque directement dans la fonction test-unitaire2 afin de générer les 4 masques successivement, avec enregistrement du score

Nous pouvons maintenant récupérer la valeur de notre score, et le masque utiliser nous permettant d’obtenir se score, pour générer le qrcode avec.

Une image contenant texte

Description générée automatiquementVoici le résultat final avec le masque optimal.

Et voici le rendu final après exportation en image

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55 | **void** **test\_unitaire\_sujet2**(**void**)  {  printf("Void test\_unitaire\_sujet2**\n\n\n**");  **unsigned** **char** MicroQRcode[NB\_MODULE][NB\_MODULE]; // le microQRcode sans la Quiet ZONE  **unsigned** **char** MicroQRmask[NB\_MODULE][NB\_MODULE]; // le mask  **unsigned** **short** **int** mode\_microQRcode = M4\_L ; // choix du mode  **unsigned** **short** **int** mask\_number = **0**; // choix du n° de MASK  **int** score, score\_max =**0**, mask; //Initialisation des  //fonctions score, score\_max et mask  efface\_QRcode(MicroQRmask); //Effacement du MicroQRmask afin  //de pouvoir en généré un sans problème de génération  printf("**\n\n\n**\*\*\*\*\*\*\*\*\*\*Selection du meilleur mask\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***\n\n\n\n**");  **for**(mask\_number=**0**; mask\_number<=**3**; mask\_number++) //Boucle permettant de générer  {  //tout les masques avec leurs QRCODES afin de sélectionner le mask meilleur mask.  genere\_QRmask(MicroQRmask,mask\_number);  efface\_QRcode(MicroQRcode);  initialise\_QRcode(MicroQRcode);  xor\_QRcode\_QRmask(MicroQRcode, MicroQRmask);  printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Score masque numéro %d\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***\n\n\n**", mask\_number);  //Affichage des différents scores de chaque masque.  score = score\_masquage\_QRcode(MicroQRcode);  printf("**\n**score du masque %d : **\n**", score);  //Maintenant réalisons une boucle mémorisant le score le plus grand  **if**(score>=score\_max)  score\_max = score;  //Réalisons une boucle mémorisant le numéro de mask en fonction  //du score  **if**(score\_max == score)  mask = mask\_number;  printf("**\n**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Meilleur score pour le masque le plus adapté");  printf("**\n\n**score max = %d et meilleur mask = %d", score\_max, mask);  printf("**\n\n\n\n**");  }  //Maintenant génération du QRCODE avec le meilleur mask  printf("**\n\n\n**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Selection du meilleur mask\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  printf("**\n\n**\*\*\*\*\*\*\*\*\*\*\*Donnee Utile\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  printf("**\n**Meilleur mask : %d **\n**Score atteint : %d", mask, score\_max);  printf("**\n\n\n**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Génération QRCODE final\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  genere\_QRmask(MicroQRmask,mask);  printf("**\n\n\n**\*\*\*\*\*\*\*\*\*\*\*\*\*Affichage QRCODE sur la console\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***\n**");  QRcode\_to\_console(MicroQRmask);  efface\_QRcode(MicroQRcode);  initialise\_QRcode(MicroQRcode);  xor\_QRcode\_QRmask(MicroQRcode, MicroQRmask);  QRcode\_to\_console(MicroQRcode);  printf("**\n** Test unitaire 2 : Export du microQRcode dans images/Mon\_premierQRcode.pgm**\n**");  QRcode\_to\_pgm(MicroQRcode,"Images/Mon\_premierQRcode.pgm");  } |